**Higley Project Report**

**Problem Statement**

The Higley Unified School District, a rapidly growing educational institution in Arizona, comprises a consortium of 16 schools catering to the Gilbert and Queen Creek areas. With an expanding student population of over 13,000, accurately predicting school enrollments becomes paramount for effective planning and resource allocation for each new academic year.

Accurate enrollment prediction enables the district to proactively plan for essential resources, such as hiring new teachers and developing adequate infrastructure. By anticipating student numbers, Higley can avoid the negative consequences associated with deviations between projected and actual enrollments. Higley previously collaborated with a vendor who provided static enrollment predictions using Excel, resulting in an accuracy rate of only 80%. These predictions were limited to one-time use, necessitating the procurement of fresh predictions each year. Consequently, this approach incurred additional costs of approximately $20,000. Moreover, the inability to update or adjust predictions based on changing circumstances imposed significant constraints on the district's operational flexibility.

In instances where actual enrollments fall below projected numbers, overbudgeting occurs, leading to wasted resources and inefficient allocation of funds. The repercussions include maintaining small class sizes, which may not be cost-effective, and potential layoffs due to surplus staff. Conversely, when actual enrollments surpass projected figures, underbudgeting becomes a concern. The district may face a shortage of resources, resulting in overcrowded classrooms and compromised educational experiences for students.

To address the limitations posed by the previous vendor's static predictions and enhance their ability to forecast enrollment accurately, Higley Unified School District seeks an advanced solution that offers flexibility, higher accuracy, and the capacity for iterative predictions. By doing so, Higley aims to optimize resource allocation, avoid unnecessary expenses, and ensure an optimal learning environment for its students.

**Solution outcome:**

The Arizona State University Cloud Innovation Center has developed a cutting-edge machine learning-based prediction engine that surpasses customer expectations by delivering an accuracy rate exceeding 90% with a deviation of less than 2% (customer requirement was a deviation less than 10%). The cost of implementing this solution is estimated to be approximately $50 per month.

The solution includes a state-of-the-art web portal that allows users to effortlessly upload data and view prediction results in a user-friendly interface. With this advanced system in place, Higley Unified School District gains the ability to generate up-to-date predictions dynamically, ensuring the accuracy and relevance of forecasted enrollment figures.

By leveraging this state-of-the-art solution, Higley Unified School District can effectively optimize their budget and resource allocation while ensuring an optimal learning environment for their students. The affordable cost of $50 per month makes this solution both accessible and cost-effective, providing Higley with a valuable tool for strategic decision-making and efficient management of their educational system.

**High-Level Architecture**

A screenshot of a computer

Description automatically generated

* At a high level, the Higley Enrollment Prediction system offers users access to a dedicated website, providing two primary use cases: model generation and inference/viewing the result.
* To generate a model, the user interacts with the system by uploading the required datasets and initiating the automated modeling process. Upon initiating the process, the user receives an email notification confirming that the automation has commenced, and the uploaded datasets are first processed by the pre-processing engine.
* Once the pre-processing stage is complete, the cleaned data proceeds to the modeling process, where the necessary algorithms and techniques are applied to generate accurate predictions. The resulting predictions are then fed into a comprehensive dashboard. Simultaneously, the user is notified of the completion of the modeling process.
* To access the updated and latest results, the user utilizes the portal to navigate to the dashboard. The dashboard provides a comprehensive view of the predictions, enabling users to make informed decisions based on the most recent insights and trends.
* This high-level architecture ensures a seamless and efficient flow of data from dataset upload to preprocessing, modeling, and final result visualization. It empowers users to leverage the system's capabilities to generate accurate enrollment predictions and access real-time information through the intuitive dashboard, enhancing their ability to make informed decisions regarding resource allocation and planning within Higley Unified School District.

**Technical Architecture :**

A screenshot of a computer

Description automatically generated

* The technical architecture of the Higley Enrollment Prediction system comprises two main phases: model generation and result inference, each with distinct components and processes.
* To provide users with access to the system, the Higley Enrollment Prediction website is developed using AWS Amplify and ReactJS, ensuring a responsive and user-friendly interface.
* During the model generation phase, the user uploads datasets via the website's user interface. The uploaded files are stored in an S3 bucket, triggering an automatic Lambda function. This Lambda function initiates a sequence of Glue ETL (Extract, Transform, Load) jobs to preprocess and integrate the raw datasets, resulting in clean and consolidated data.
* The preprocessed data is then stored back in the same input S3 bucket, ready for use in the modeling phase. Triggers within the system activate a Sagemaker instance, which, in turn, triggers the Sagemaker lifecycle configuration. The configuration runs Sagemaker notebook cells sequentially, executing the modeling process. The resulting model outputs are stored in an S3 output bucket for further analysis and reference.
* Additionally, a preprocessing engine is triggered to run Glue crawlers over the datasets, generating metadata. This metadata is then utilized to execute an Athena query, which serves as input for the Quicksight dashboard.
* To access the Quicksight dashboard, users can click the designated button on the Higley Enrollment Prediction website, redirecting them to the actual dashboard interface. This allows users to interact with the dashboard, visualizing the modeling results and gaining valuable insights.
* The technical architecture described above ensures a seamless flow of data from dataset upload to preprocessing, modeling, and result visualization. Leveraging AWS services such as S3, Lambda, Glue, Sagemaker, Athena, and Quicksight, the system offers efficient data processing, modeling capabilities, and intuitive dashboard access to facilitate accurate enrollment predictions for the Higley Unified School District.

**Project Flow and Approach**

A diagram of a model

Description automatically generated with low confidence

In the modern era, where data plays a central role, machine learning has emerged as a potent technique for extracting valuable insights and making accurate predictions from vast and intricate datasets. At the core of this process lies the machine learning pipeline (MLP), which comprises a sequence of interconnected steps designed to convert raw data into a trained and deployed machine learning model. In this presentation, we will delve into the distinct phases of an MLP and underscore their critical importance in constructing resilient and impactful machine learning solutions.

1. Pre-processing:

Non-NDA Dataset Integration:

To create a comprehensive non-disclosure agreement (NDA) dataset, we integrated publicly available datasets such as land development, census, and housing data with NDA-protected school enrollment and school lunch data. This integration allowed us to gain a holistic understanding of factors influencing student outcomes.

NDA Dataset Utilization:

Academic Activities Dataset:

We utilized an academic activities dataset that provided a tabulated representation of students' activities for each academic school year. This dataset helped us analyze the impact of extracurricular involvement on student performance and engagement.

Residency Address Dataset:

Another essential dataset we leveraged was the residency address dataset, which contained student addresses along with start and end dates. This information enabled us to explore the relationship between student demographics and academic outcomes, considering factors like neighborhood characteristics and stability.

Streamlined Student Enrollments:

Ensuring data quality, we addressed duplicate records and occurrences of students within a single school year. We identified and eliminated duplicate entries, reducing the dataset size by approximately 12%. This process accounted for student transfers between schools, enabling us to focus on unique student experiences.

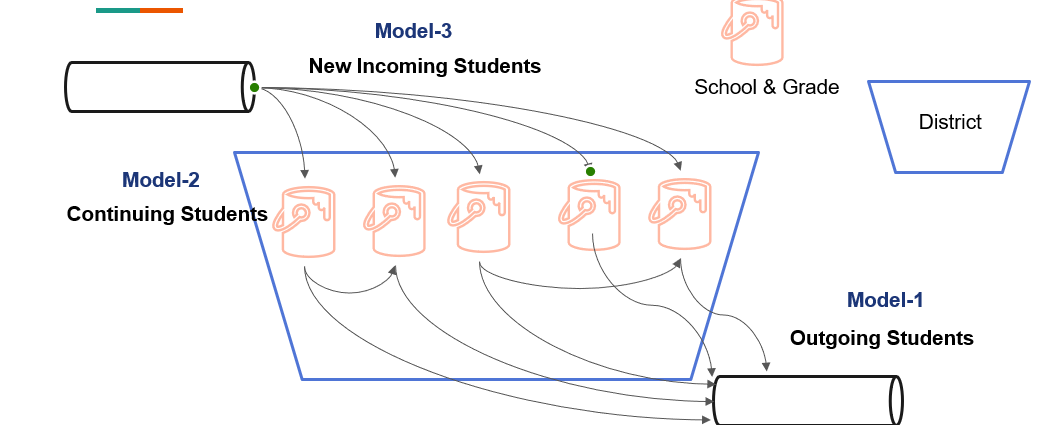
Historical New Intakes Table:

To analyze trends and changes over time, we created a comprehensive table called the Historical New Intakes Table. This table documented the total number of new intake students, categorized by school and grade, across multiple years. Analyzing this data helped identify enrollment patterns and evaluate the effectiveness of implemented programs and initiatives.

By integrating and pre-processing these diverse datasets, we established a solid foundation for our machine learning models. This ensured that our analysis was based on reliable and relevant information, empowering us to make informed decisions and draw meaningful conclusions in the subsequent stages of our project.

1. Modelling:

**MLP - Enrollment (Bucket enrollment prediction model)**



The Modelling phase consists of three key steps: Model Training and Evaluation, Model Optimization and Validation, and Model Deployment and Monitoring.

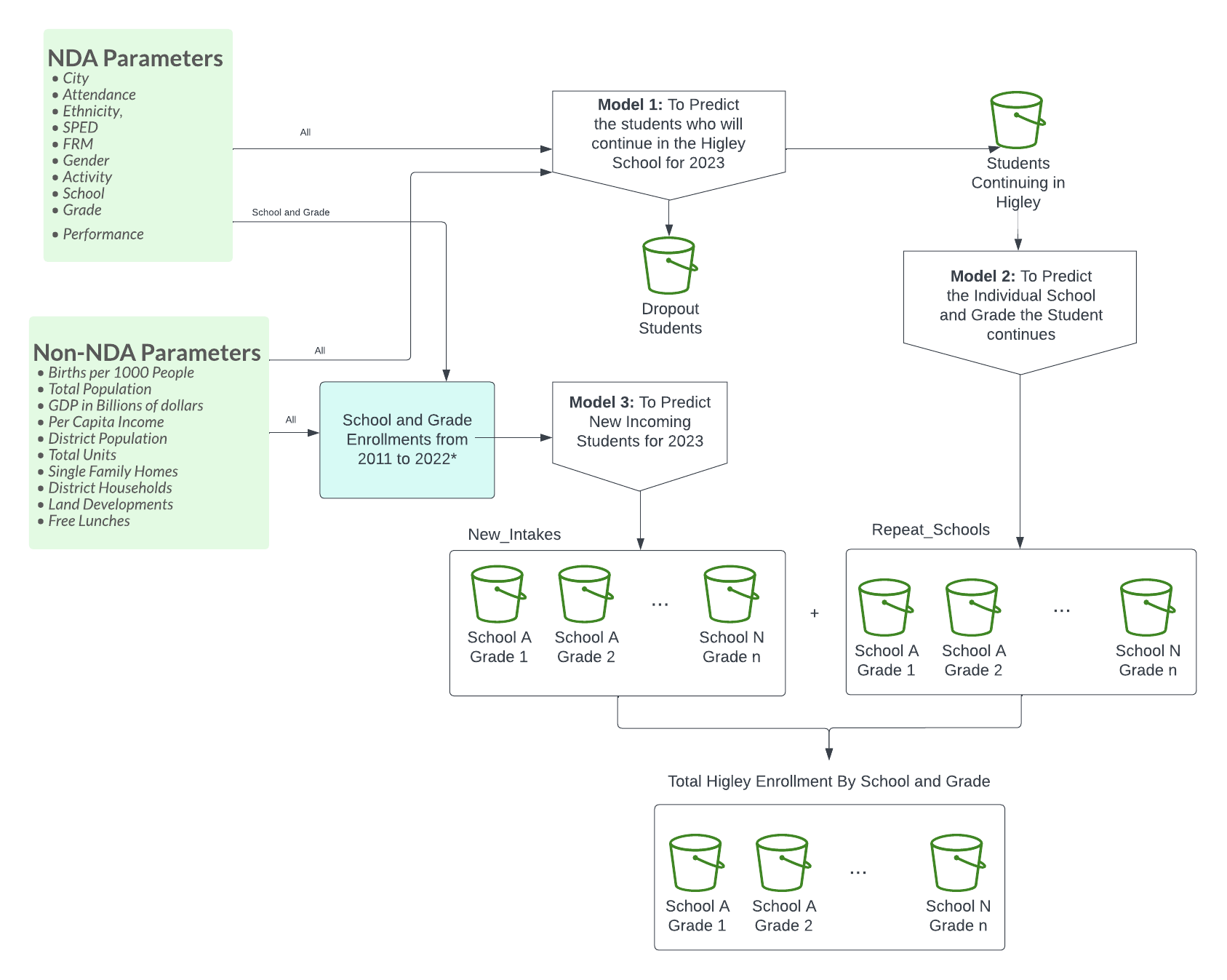
During Model Training and Evaluation, the prepared data is used to train a machine learning model. Various algorithms are applied based on the nature of the problem. The model is trained using a portion of the data, and its performance is assessed. To enhance the model's performance, techniques such as hyperparameter tuning and optimization are employed. This involves adjusting the model's parameters to find the optimal configuration that maximizes its performance on the validation dataset. Cross-validation techniques ensure the model's generalizability and its ability to perform well on unseen data.

Once the model is optimized and validated, it is ready for deployment. In our use case, we developed the Bucket enrollment prediction model, which combines three different models:

* Model-1 predicts students who will repeat a grade in the Higley district in the following year, as well as those who are likely to drop out.
* Model-2 predicts whether repeating students will continue in the same school within the Higley district or transfer to a new school.
* Model-3 predicts the total number of new students entering the Higley school district based on historical trends.

By using this combined approach, we capture the dynamic flow of students within the district, accounting for factors that influence movements and enrollment changes.

**Enrollment Prediction Models (integrated view)**



1. Post Processing:

As mentioned, in this stage we refresh the datasets used by the dashboard with the model prediction outputs. This allows us to incorporate the latest predictions generated by the models into the datasets, ensuring that the dashboard presents the most up-to-date information. By integrating the predictions, we enhance the accuracy and relevance of the data presented in the dashboard.

In addition to refreshing the datasets, we have implemented an automated email notification system. This system plays a crucial role in communicating with the users and keeping them informed about the completion of the modeling process. Once the models have finished processing the data and generating predictions, an email is automatically sent to the user. This email serves as a notification, indicating that the modeling process is complete and a direct link to access the dashboard.

By sending this email, we ensure that the users are promptly informed about the availability of the updated predictions and can conveniently access the dashboard to explore the results. This streamlines the communication process and allows users to stay engaged with the project and make timely decisions based on the latest insights.

**Project Outcome:**

The Arizona State University Cloud Innovation Center (ASU CIC) collaborates with the Higley School District to develop an AWS-based prototype tool aimed at achieving the following objectives:

* Utilizing Machine Learning techniques to predict future enrollment.
* Integrating real-time data on an iterative rolling basis to ensure up-to-date predictions.
* Providing comprehensive analytics through a user-friendly dashboard.

To train the dataset, the XGBoost model is employed. XGBoost is a widely adopted and efficient open-source implementation of the gradient boosted trees algorithm. This algorithm is designed to be flexible, portable, and highly efficient. It combines the estimates of multiple simpler models to accurately predict the target variable in a supervised learning setting.

MODEL - 1 (STUDENT DROPOUT)

The first model is a classifier that utilizes historical data containing both NDA and non-NDA information to categorize students as repeating (1) or non-repeating (0). The "REPEAT" attribute is added as the output parameter, where a value of 1 indicates the student continues in the Higley school district, while 0 denotes dropout. This model allows us to identify the number of students repeating in the next academic year and their respective grade counts.

MODEL - 2 (REPEATING STUDENTS)

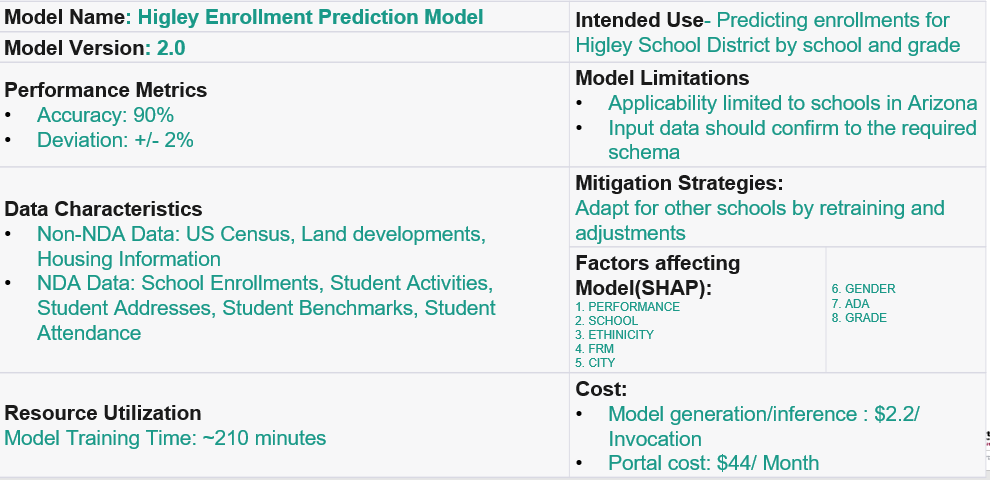
The second model is a classifier focused on repeating students. It considers student-related information and non-NDA details to determine whether a repeating student will continue in the same school (0) or transfer to a different eligible school within the Higley district (1). The "REPEAT\_SCHOOL" attribute is added as the output parameter, providing insights into whether students repeat in the same school or move to another based on their behavior patterns. If a student repeats in the same school, they are added to the corresponding school and grade bucket. Otherwise, a separate classification model is employed to predict the most probable school the student will attend based on historical data.

MODEL - 3 (NEW STUDENTS)

The third model employs regression techniques to predict the total number of new students that will join the Higley schools in the next academic year. This prediction is based on historical data regarding new intakes and school enrollments. By using this model, we can fill the buckets with all the students who will join the schools in the upcoming year.

To enhance the model's accuracy, an iterative training process is adopted. The current prediction results serve as training data for future enrollment predictions. By incorporating new information and retraining the model on the complete dataset, the accuracy of predictions strengthens over time. This iterative training approach ensures that the model continuously adapts to the changing enrollment patterns and improves its predictive capabilities, providing more accurate forecasts to support the planning and decision-making processes within the Higley School District.

**Model Card:**



**Model Test Results(2023):**

ACTUAL SCHOOL DATA FOR 2023:

A screenshot of a table

Description automatically generated with low confidence

PREDICTED SCHOOL ENROLLMENT FOR 2023:

A screenshot of a table

Description automatically generated with low confidence

**Monthly Cost:**

Given below is the total cost associated if the model is generated once:

\*AWS COST AS OF 6/20/23

|  |  |
| --- | --- |
| **AWS RESOURCE** | **COST** |
| AWS LAMBDA | $0.1 |
| AWS GLUE ETL | $0.95 |
| AWS SAGEMAKER | $0.82 |
| AWS SES | $0.01 |
| AWS AMPLIFY | $0.3 |
| AWS S3 | $0.02 |
| AWS GLUE CRAWLER | $0.03 |
| AWS ATHENA | $0.01 |
| AWS QUICKSIGHT | $44 |
| **TOTAL** | **$46.24** |

**Issues/Assumptions:**

A screenshot of a computer

Description automatically generated with low confidence

A picture containing text, screenshot

Description automatically generated

**User Documentation / Instructions to run**

1. Access the Higley School District website.

A screenshot of a login

Description automatically generated with medium confidence

1. If you already logged in previously, sign in with your credentials or else signup as follows:

A screenshot of a computer screen

Description automatically generated with low confidence

1. Once you log in into the homepage, it looks like this:

A room with desks and chairs

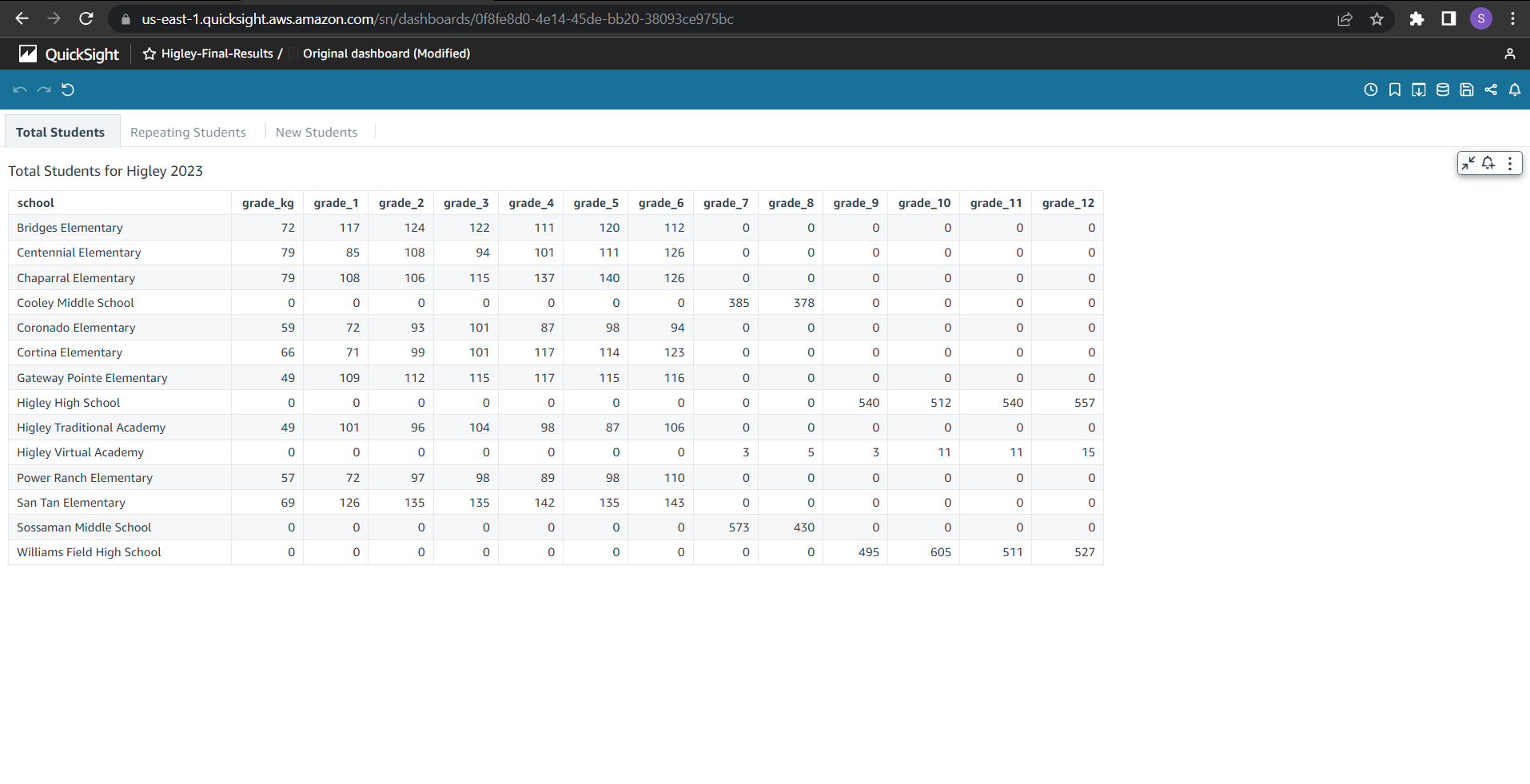
Description automatically generated with low confidence

1. Click on start prediction to proceed to the next screen.
2. If you would like to visit an already made prediction for the next year, click Yes else click No below:

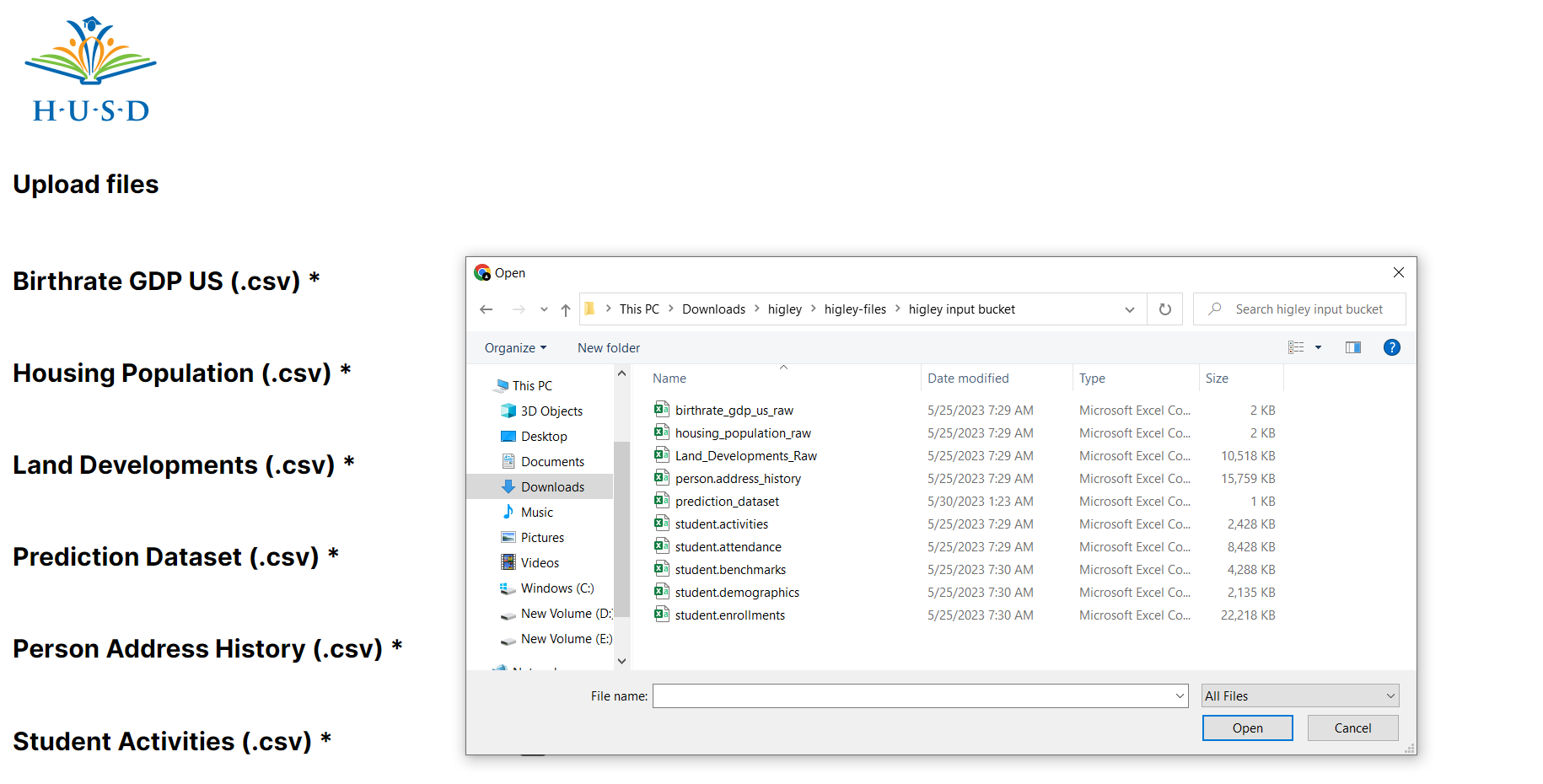
A screenshot of a computer

Description automatically generated with medium confidence

1. If you click yes, you will be redirected to the AWS Quicksight dashboard displaying the results of Total Students, Repeating Students and New Intake student by School and Grade as follows:



1. If no is clicked, you can upload the datasets in the next screen:



1. Once the datasets are uploaded, click continue and the automated modelling starts and you will be notified to your registered email.

A screenshot of a computer

Description automatically generated with low confidence

1. Kindly wait for the modelling process to finish, which takes approximately 210 minutes to complete. Later you will be notified through email on completion so you can infer the results from the web app.

A screenshot of a chat

Description automatically generated with low confidence

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**AWS Account Setup Instructions**

S3 BUCKETS:

1. Create the following S3 Buckets (using default configurations):
   1. [higley-temporary-bucket](https://s3.console.aws.amazon.com/s3/buckets/higley-temporary-bucket?region=us-east-1)( potential feature to be improved)
   2. [higley-input-bucket](https://s3.console.aws.amazon.com/s3/buckets/higley-input-bucket?region=us-east-1)
   3. [higley-output-bucket](https://s3.console.aws.amazon.com/s3/buckets/higley-output-bucket?region=us-east-1)
2. Make the higley-temporary bucket publicly accessible with the following:
   1. BUCKET POLICY: (Copy from higley-temporary-bucket-policy)
   2. CORS: (Copy from higley-temporary-bucket-cors)
3. Copy the txt files current\_user.txt, last\_model.txt from backend-> other folder to the s3-input-bucket

LAMBDA FUNCTIONS:

1. Create higley-bucket-transfer lambda function with author from scratch option and select Runtime as Python 3.10 and other options default.The lambda function higley-bucket- is triggered on upload to higley-temporary-bucket and uploads to higley-input-bucket.
   1. Select Add trigger on the screen and select s3 source and select [higley-temporary-bucket](https://s3.console.aws.amazon.com/s3/buckets/higley-temporary-bucket?region=us-east-1)
   2. Select all object create events, acknowledge and confirm.
   3. Set the general configuration as follows:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. In IAM settings, go to the role associated with the lambda function and attach Amazon S3 Full Access:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Copy the code for the lambda function from GitHub.

1. Create trigger-verification-lambda with author from scratch option and select Runtime as Python 3.10 and other options default as follows:
   1. Set the general configuration as:

A screenshot of a computer

Description automatically generated with low confidence

* 1. For the Lambda function role, attach the following policies:

A screenshot of a computer

Description automatically generated

* 1. Also create a new policy and attach. The Json for the newly created lambda\_all is in the backend->lambda\_all.txt
  2. Modify sender email address in the code.

1. Create higley-transformations-lambda with author from scratch option and select Runtime as Python 3.10 and other options default as follows:
   1. Set the general configuration as follows:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Set the following IAM policies:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Copy the code for the lambda function from github.

1. Create higley-cleaning-lambda with author from scratch option and select Runtime as Python 3.10 and set the IAM policy as of higley-transformations-lambda and the other setup as :
   1. Set the general configuration as follows:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Copy the code from github

1. Create bottomup-prediction with author from scratch option and select Runtime as Python 3.10 and configure the rest as:
   1. Set the general configuration as follows:

A picture containing text, screenshot, font, line

Description automatically generated

* 1. Set the following IAM policies:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Copy the code from github and modify sender, aws\_account\_id in the code.

GLUE ETL JOBS:

Create Python Shell script editor ETL Scripts and select Create a new script with boilerplate code.

1. [activities-transformation](https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/activities-transformation)
2. [address-transformation](https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/address-transformation)
3. [student-enrollments](https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/student-enrollments)
4. [Student-Enrollment-Transformation](https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/Student-Enrollment-Transformation)
5. [non-nda transformation](https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/non-nda%20transformation)
6. [new-intakes transformation](https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/new-intakes%20transformation)
7. [cleaning-1](https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/cleaning-1)
8. [cleaning-2](https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/cleaning-2)

Attach s3 full acess permission to all etl jobs and extra sagemaker full access permission to cleaning-1.

**SAGEMAKER:**

1. Go to Sagemaker Lifecycle configuration and create a configuration under notebook instances and set to run on start.
2. Create a Sagemaker Instance [Higley-Bottomup](https://us-east-1.console.aws.amazon.com/sagemaker/home?region=us-east-1#/notebook-instances/Higley-Bottomup) and use ml.c5.2xlarge instance.
3. Upload the bottomup-model.ipynb to the sagemaker instance
4. Stop the instance.
5. Once stopped, Attach the sagemaker lifecycle configuration in sagemaker-lifecycle-configuration.txt and do not start.
6. Also attach the following policies to the sagemaker notebook instance role:

A screenshot of a computer

Description automatically generated with medium confidence

GLUE CRAWLER:

1. Create the following databases in glue databases:

A picture containing text, screenshot, font, line

Description automatically generated

1. Next Create Tables within the database using the csv files inside the Higley-output-bucket as:

A picture containing text, font, line, number

Description automatically generated

1. The first column is the table name , the second column is the db name and the third column is the corresponding s3 path. Use Default Settings for the others.
2. Create a Glue Crawler that goes through the csv files and updates the Glue tables within the database as follows:
   1. Visit the Glue crawlers page and create the following:

A screenshot of a computer

Description automatically generated with low confidence

* 1. For each of the crawler, select that the crawler has not yet been mapped to any glue tables.
  2. Also set the data source as s3 and set the path of the corresponding s3 folder and select the following option to crawl:

A screenshot of a computer

Description automatically generated with low confidence

* 1. Consider the following:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

ATHENA:

1. Query the Database tables running a general Athena query as “Select \* from <table-name>”
2. Run for all the tables new\_intake\_students, repeating\_students and total\_students.

QUICKSIGHT:

1. Go to datasets in Quicksight and create new datasets.
2. Select Athena and give some name to the data source name and Athena group to primary.

A screenshot of a computer

Description automatically generated

1. Validate connection and create the data source.
2. Repeat the process to create 3 Athena data sources to attach to the quicksight dashboard:

A picture containing text, font, line, screenshot

Description automatically generated

1. Create Analysis to display tables for Total Students, Repeating Students and New Intake Students as follows using table icon on new visualization and drag to the analysis page:

A screenshot of a computer

Description automatically generated

1. Set the source for each of the table as Athena and map the corresponding data sources
2. Once the analysis is ready, publish the dashboard through share option at the top right:

A screenshot of a computer

Description automatically generated with medium confidence

1. Save the name of the new dashboard as Higley-Final-Results.
2. Copy the url and hardcode the url in the global variables of the frontend code in frontend->src->pages->Globalvariables.js

**FRONTEND-AMPLIFY:**

1. Open AWS Amplify Console by searching through AWS Services.

2. Clone the code to your own GitHub repository

3. Click on New App -> Host web app.  
4. Select GitHub as the source of Existing Code   
5. Connect to the repository with branch as ‘main’ and put ‘frontend’ as the directory name since repository is a monorepo  
6. Create a new environment.  
7. Create or use a service role that has Amplify permissions.  
8. In build and test settings, add the following lines of code under build-  
>commands. send env vars  
```  
 build:  
 commands:  
 - REACT\_APP\_ACCESS\_ID=${REACT\_APP\_ACCESS\_ID}  
 - REACT\_APP\_ACCESS\_KEY=${REACT\_APP\_ACCESS\_KEY}

- REACT\_APP\_REGION =${ REACT\_APP\_REGION }   
 - npm run build  
```  
9. In advanced permissions, set environment variables with your credentials  
a. REACT\_APP\_ACCESS\_ID  
b. REACT\_APP\_ACCESS\_KEY

c. REACT\_APP\_REGION  
(Note: These credentials should have read access to the S3 bucket   
where files have to be uploaded and write access to the interim S3   
bucket )

10. Change configurations as per your requirement and start the build thereby generating the amplify url.

DATASETS:

The data to be uploaded needs to be in a certain schema which can be found in backend->other->data\_schema.xlsx

CHEAT SHEET FOR SETUP:

Given below is the order to setup the resources from github on your aws account:

BACKEND:

1. Create higley-temporary-bucket and configure its bucket policy and cors

2. Create higley-input-bucket and copy 2 empty files current\_user.txt, last\_model.txt

3. Create lambda function higley-bucket-transfer that is triggered on upload to higley-temporary-bucket and verify the upload. Replicate its permissions and configurations

4. Create SES Identity for sender email

5. Create trigger-verification-lambda and replicate its permissions and configurations. Modify sender email address.

6. Create higley-transformations-lambda and replicate its permissions and configurations

7. Create higley-cleaning-lambda and replicate its permissions and configurations

8. Create glue python etl jobs for 'activities-transformation', 'address-transformation', 'student-enrollments', 'Student-Enrollment-Transformation', 'non-nda transformation', 'new-intakes transformation','cleaning-1' and 'cleaning-2' and provide required permissions to each of them

9. Create higley-output-bucket.

10. Test Data Engineering lifecycle by commenting sagemaker instance on in cleaning-1.Run trigger-verification-lambda passing testing email-id as payload.

11. Create sagemaker instance Higley-Bottomup(ml.c5.2xlarge) and sagemaker notebook bottomup-model thereby attach the sagemaker configuration sagemaker-lifecycle-configuration. Give necessary permissions to the sagemaker role.

12. Create lambda function bottomup-prediction and modify sender, aws\_account\_id details

13. Create Glue crawlers, databases and tables replicating the existing configurations

14. Create athena queries over the datasets and store the results in higley-athena-results

15. Create a quicksight dashboard and create athena query datasets with names total\_students, repeating\_students and new\_intake\_students.

16. Attach the athena datasets to quicksight dashboard

17. Publish the dashboard and copy the url.

FRONTEND:

1. Make code changes in the Global variables page

2. Deploy amplify build as per requirement